AXIOMATIC
Global Electronic Solutions

Q&A - What is CiA CANopen?

[Source for following answers: http://www.can-cia.org/canopen/]

What is CiA CANopen?

CANopen is a CAN-based higher layer protocol. It was developed as a standardized embedded network with highly flexible configuration capabilities. CANopen was designed for motion-oriented machine control networks, such as handling systems. By now it is used in many various fields, such as medical equipment, off-road vehicles, maritime electronics, public transportation, building automation, etc.

The CANopen application layer and communication profile (EN 50325-4; CiA 301) supports direct access to device parameters and transmission of time-critical process data. The CANopen network management services simplify project design, system integration, and diagnostics. In each decentralized control application, different communication services and protocols are required. CANopen defines all these services and protocols as well as the necessary communication objects.

Any CANopen device can be seen as a generic device. This generic device is connected to CAN on one side and connected to application specific I/O data on the other side. The application is the key knowledge of the device manufacturer. The interface between the application and CANopen is realized by the object dictionary. The object dictionary is unique for any CANopen device and represents the whole access to its implemented application in terms of data as well as in terms of configuration. To gain access to the object dictionary each CANopen device has to realize a CANopen protocol stack. This CANopen protocol stack is a piece of software, which normally is implemented on the same controller that is used by the application software.

The CANopen protocol stack consists of different functions for different purposes.
Process Data Object (PDO) is used to transmit the application data. The application data is transmitted without any protocol overhead in broadcast.
Service Data Object (SDO) is used to gain access to all device parameters. SDO is used for direct device-to-device communication.
Error Control is used to validate that any device is working proper in terms of CANopen communication.
Network Management is used to control the network in terms of CANopen communication and indirectly in terms of system behaviour.

What is an Object Dictionary?

The Object Dictionary describes the complete functionality of a device by way of communication objects and is the interface between the communication interface and the application program. All communication objects of a device (application data and configuration parameters) are described in its Object Dictionary in a standardized way. These objects are accessible by a 16-bit index and in the case of arrays and records there is an additional 8-bit sub-index.

The Object Dictionary is broken into four areas: Communication Profile (Index 1000h to 1FFFh); Manufacturer Specific (Index 2000h to 5FFFh); Standardized Device Profile (6000h to 9FFFh); and Standardized Interface Profile (Index A000h to BFFFh).

The Communication Profile Area contains the communication specific parameters for the CAN network. The Standardized Device Profile Area contains all data objects common to a class of devices that can be read or written via the network. The device profiles may use these entries to describe the device parameters and device functionality. The Manufacturer Specific Area is left in for specific functionality that is not covered by the standard device profile.

What kinds of messages are sent on a CANopen network?

CANopen uses an 11-bit identifier called a COB-ID. The upper four bits if the COB-ID is the function code of the subsequent 8-byte message, and the lower 7 bits are the Node-ID of the transmitting device. Broadcasting of non-confirmed NMT-, SYNC- and TIME-STAMP-objects is indicated by a Node-ID of zero.

There is a pre-defined connection set which supports one emergency object, one SDO (tx), one SDO (rx), a maximum of four received PDOs (RPDO), a maximum of four transmit PDOs (TPDO), and the NMT objects.

Broadcast Objects of the Pre-define Connection Set

Object	**Fiction Code (binary)**	**Resulting COB-ID**
NMT	0000	0
SYNC	0001	128 (80h)
TIME STAMP	0010	256 (100h)

Peer-to-Peer Objects of the Pre-define Connection Set

Object	**Fiction Code (binary)**	**Resulting COB-IDs**
EMERGENCY	0001	129 (81h) to 255 (FFh)
PDO1 (tx)	0011	385 (181h) to 511 (1FFh)
PDO1 (rx)	0100	513 (201h) to 639 (27Fh)
PDO2 (tx)	0101	641 (281h) to 767 (2FFh)
PDO2 (rx)	0110	769 (301h) to 895 (37Fh)
PDO3 (tx)	0111	897 (381h) to 1023 (3FFh)
PDO3 (rx)	1000	1025 (401h) to 1151 (47Fh)
PDO4 (tx)	1001	1153 (481h) to 1279 (4FFh)
PDO4 (rx)	1010	1281 (501h) to 1407 (57Fh)
SDO (tx)	1011	1409 (581h) to 1535 (5FFh)
SDO (rx)	1100	1537 (601h) to 1663 (67Fh)
NMT Error Control	1110	1793 (701h) to 1919 (77Fh)

What is an NMT Object?

The Network Management (NMT) Objects include the Boot-up message, Heartbeat protocol, and NMT message. The Boot-up message, and Heartbeat protocol are implemented as single CAN frames with a 1-byte data field.

A device sends the Boot-up message to indicate to the NMT master that it has reached the state Pre-operational. This occurs whenever the device initially boots-up but also after a power-out during operation. The Boot-up message has the same identifier as the Heartbeat object, but its data content is zero.

The Heartbeat protocol is for error control purposes and signals the presence of a node and its state. The Heartbeat message is a periodic message of the node to one or several other nodes. It indicates that the sending node is still working properly. Besides Heartbeat protocol there exists an old and out-dated error control services, which is called Node and Life Guarding protocol. However, it is not recommended for implementation.

The NMT message is mapped to a single CAN frame with a data length of 2 bytes. (The COB-Identifier is 0). The first byte contains the command specifier and the second contains the Node-ID of the device that must perform the command (in the case of Node-ID 0 all nodes have to perform the command). The NMT message transmitted by the NMT master forces the nodes to transit to another NMT state.

The CANopen state machine specifies the states Initialization, Pre-Operational, Operational and Stopped. After power-on, each CANopen device is in the state Initialization and automatically transits to the state Pre-operational. In this state, transmission of SDOs is allowed. If the NMT master has set one or more nodes into the state Operational, they are allowed to transmit and to receive PDOs. In the state Stopped no communication is allowed except that of NMT objects.

The state Initialization is divided into three sub-states in order to enable a complete or partial reset of a node. In the sub-state Reset Application the parameters of the manufacturer-specific profile area and the standardized device profile area are set to their power-on values. In the sub-state Reset Communication the parameters of the communication profile area are set to their power-on values. The third sub-state is initializing, which a node enters automatically after power-on. Power-on values are the last stored parameters.

What is an SDO?

A Service Data Object (SDO) is used to read entries from, or write entries to, a module's object dictionary. The SDO transport protocol allows transmitting objects of any size. The first byte of the first segment contains the necessary flow control information including a toggle bit to overcome the well-known problem of doubly received CAN frames. The next three bytes of the first segment contain index and sub-index of the Object Dictionary entry to be read or written. The last four bytes of the first segment are available for user data. The second and the following segments (using the very same CAN identifier) contain the control byte and up to seven bytes of user data. The receiver confirms each segment or a block of segments, so that a peer-to-peer communication (client/server) takes place.

What is a PDO?

A Process Data Object (PDO) is mapped to a single CAN frame using up to 8 bytes of the data field to transmit application objects. Each PDO has a unique identifier and is transmitted by only one node, but it can be received by more than one node (producer/consumer communication). PDO transmission is not confirmed.

PDO transmissions may be driven by an internal event, by an internal timer, by remote requests and by receiving the SYNC message. An internal event that triggers message transmission is specified in the device profile, as would be an elapsed timer that would trigger periodic transmissions of the PDO. Another device may initiate the transmission of an asynchronous PDO by sending a remote transmission request (remote frame).

In order to initiate simultaneous sampling of input values of all nodes, a periodically transmitted SYNC message is required. Synchronous transmission of PDOs takes place in cyclic and acyclic transmission mode. Cyclic transmission means that the node waits for the SYNC message, after which it sends its measured values. Its PDO transmission type number (1 to 240) indicates the Sync rate it listens to (how many SYNC messages the node waits before the next transmission of its values). Acyclically transmitted synchronous PDOs are triggered by a defined application-specific event. The node transmits its values with the next SYNC message but will not transmit again until another application-specific event has occurred.

The default mapping of application objects, as well as the supported transmission mode(s), are described in the Object Dictionary for each PDO. PDO identifiers should have high priority to guarantee a short response time.

The PDO mapping defines which application objects are transmitted within a PDO. It describes the sequence and length of the mapped application objects. A device that supports variable mapping of PDOs must support this during the pre-operational state. If dynamic mapping during operational state is supported, the SDO Client is responsible for data consistency.

What are Special Function Objects?

CANopen also defines three specific protocols for synchronization, emergency indication, and time-stamp transmission.

The Synchronization (SYNC) Object is periodically broadcast by the SYNC Producer. The time period between SYNC messages is defined by the Communication Cycle Period, which may be reset by a configuration tool to the application devices during the boot-up process. There can be a time jitter in transmission by the SYNC Producer due to some other objects with higher prior identifiers or by one frame being transmitted just before the SYNC message. The SYNC message is mapped to a single CAN frame with the identifier 128 by default and does not carry any data.

The Emergency (EMCY) messages are triggered by the occurrence of device internal errors, and are transmitted from an Emergency producer on the concerned application device. This makes them suitable for interrupt type error alerts. An Emergency message is transmitted only once per ‘error event’. As long as no new errors occurs on a device, no further Emergency message can be transmitted. Zero or more Emergency consumers may receive these. The reaction of the Emergency consumer is application-specific. CANopen defines several Emergency Error Codes to be transmitted in the Emergency message, which is a single CAN frame with 8 data bytes.

By means of Time-Stamp Object, a common time frame reference is provided to application devices. It contains a value of the type Time-of-Day. This object transmission follows the producer/consumer push model. The associated CAN frame has the pre-defined identifier 256 and a data field of 6-byte length.